ES 204 – Numerical Methods in Engineering

2^{nd} Semester AY 2017-2018

Machine Problem 01

Submitted by:

Eldion Vincent H. Bartolo

2010-28167

Master of Science in Electrical Engineering

University of the Philippines - Diliman

Calculating the Dimensions of a Uniform Fin for a Specific Thermal Effincency Through Newton-Raphson method

Problem Statement

The necessary cross-sectional dimensions of an aluminum fin with a square cross-section is to be determined such that it will have thermal efficiency of 0.95. Newton-Raphson method is used to solve the nonlinear mathematical model for thermal efficiency of a uniform fin, considering convection through its tip.

Results and Discussion

The mathematical model for the thermal efficiency of a uniform fin is described by fig. 1 wherein l - length, A - cross-sectional area, P - perimeter, k - thermal conductivity and h_∞ - heat transfer coefficient of the fin.

$$\eta = \left(\frac{\sinh \frac{l}{\lambda} + \alpha \cosh \frac{l}{\lambda}}{\cosh \frac{l}{\lambda} + \alpha \sinh \frac{l}{\lambda}} \right) \left(\frac{\lambda}{l + \frac{A}{P}} \right) \quad (1)$$

$$\lambda = \left(\frac{kA}{h_\infty P} \right)^{1/2} \quad (2)$$

$$\alpha = \left(\frac{h_\infty A}{kP} \right)^{1/2} \quad (3)$$

Figure 1: Mathematical model for the thermal efficiency of a Uniform Fin.

Considering that the values of l = 0.1m, k – 240 W/(m ·°C) and h_∞ = 9 W/(m^2 ·°C) are given and A and P are dependent on s - dimension of the side of a square, the discussed mathematical model can be expressed in terms of s only.

The mathematical model is divided into parts and the associated first derivatives with respect to s are computed. The derivation are described by the following statements.

Note that (x ') symbolizes the first derivative of the variable x with respect to s.

- Let $\eta = AB$ and $\eta' = A'B + AB'$

- $A = \dfrac{sinh\left(\frac{l}{\lambda}\right)+\alpha cosh\left(\frac{l}{\lambda}\right)}{cosh\left(\frac{l}{\lambda}\right)+\alpha sinh\left(\frac{l}{\lambda}\right)}$ and $A' = \dfrac{C'D-CD'}{D^2}$

- $B = \dfrac{\lambda}{l+\left(\frac{A}{P}\right)}$ and $B' = \dfrac{E'F-EF'}{F^2}$

- $C = sinh\left(\frac{l}{\lambda}\right) + \alpha cosh\left(\frac{l}{\lambda}\right)$ and
  $C' = \left[\, cosh\left(\frac{l}{\lambda}\right) + \alpha sinh\left(\frac{l}{\lambda}\right) \right] [-l\,\lambda^{-2}\,]\,\lambda' \;+\; cosh\left(\frac{l}{\lambda}\right)\alpha'$

- $D = cosh\left(\frac{l}{\lambda}\right) + \alpha sinh\left(\frac{l}{\lambda}\right)$ and

  $D' = \left[\, sinh\left(\frac{l}{\lambda}\right) + \alpha cosh\left(\frac{l}{\lambda}\right) \right] [-l\,\lambda^{-2}\,]\,\lambda' \;+\; sinh\left(\frac{l}{\lambda}\right)\alpha'$
- $E = \lambda$ and $E' = \lambda'$
- $F = l + \left(\frac{A}{P}\right)$ and $F' = \left(\frac{A}{P}\right)'$
- $\lambda = (\frac{k}{h})^{0.5} \left(\frac{A}{P}\right)^{0.5}$ and $\lambda' = 0.5(\frac{k}{h})^{0.5} \left(\frac{A}{P}\right)^{-0.5} \left(\frac{A}{P}\right)'$
- $\alpha = (\frac{h}{k})^{0.5} \left(\frac{A}{P}\right)^{0.5}$ and $\alpha' = 0.5(\frac{h}{k})^{0.5} \left(\frac{A}{P}\right)^{-0.5} \left(\frac{A}{P}\right)'$
- $\left(\frac{A}{P}\right) = \frac{s^2}{4s} = 0.25s$ and $\left(\frac{A}{P}\right)' = 0.25$

A C Program (*see Appendix*) is created to implement a Newton-Raphson method in solving the problem.

A function f(x) = η – 0.95 is used in the program and its first derivate with respect to *s* is computed through the derived values of A', B', C', D', E', F', λ', α', and $\left(\frac{A}{P}\right)'$.

The results of the program are shown in figs. 2-3.

Iteration	Root	Thermal_Efficiency	Absolute Error
1	0.088125489	0.991684179	0.011874511
2	0.078059757	0.990988836	0.010065732
3	0.069469660	0.990233273	0.008590097
99	0.009870963	0.950004577	0.000000120
100	0.009870855	0.950004090	0.000000107
101	0.009870759	0.950003655	0.000000096

Figure 2: First and Last 3 iterates of the NR method.

```
Converged at 0.009870759 meters, after 101 iterations

Thus the square cross section has
Side Length of:  0.009870759 meters
Area of       :  0.000097432 square meter
Perimeter of  :  0.039483037 meters
```

Figure 3: Cross-Sectional dimensions of the square fin.

The logical initial guess for *s* was 0.1 m which is the length of the uniform fin.

As shown in fig. 3, the NR method converged at a side dimension of **0.009870759 meter** after 101 iterations.

Problems encountered

Since the creation of C Program that uses NR method in root finding was already done in previous homework, the only problem encountered in solving the problem was the simplification of the thermal efficiency model and its corresponding first derivative. The most difficult part was the derivation of the first derivative because there are a multiple chain rule applications. I was also forced to review the derivatives of the hyperbolic functions and the Product and Quotient Rule for derivatives. I was able to solve these problems through research of first derivate calculations.

References

- http://mathworld.wolfram.com
- P. Dawkins, Paul's Notes – Calculus I, pp.197-198, downloadable from http://tutorial.math.lamar.edu/
- K. J. Yap, ES 204 Notes - Solution of Single Non-Linear Equations.

Appendix

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define MAXERR 0.0000001

/*  Problem # 1:
    Newton Raphson Method
*/

double therm_eff(double s)
{
    double n;
    double p1, p2, p3;
    double lambda, alpha;
    double k, l, h;
    double A, P;

    k = 240;
    h = 9;
    l = 0.1;

    A = s*s;
    P = s*4;
    lambda = sqrt((k*A)/(h*P));
    alpha = sqrt((h*A)/(k*P));

    p1 = sinh(l/lambda) + alpha*cosh(l/lambda);
    p2 = cosh(l/lambda) + alpha*sinh(l/lambda);
    p3 = (lambda/(l+(A/P)));

```

```
    n = (p1/p2)*p3;

    return n;

}
double derivative(double s)
{
    double dfx;
    double z, dz;
    double Alpha, dAlpha, Lambda, dLambda;
    double k, l, h;
    double A, B, C, D, E, F;
    double dA,dB,dC,dD,dE,dF;

    k = 240;
    h = 9;
    l = 0.1;

    z = 0.25*s;        /** Area over Perimeter **/
    dz = 0.25;         // Derivative of Area over Perimeter

    Alpha = sqrt(h/k) * sqrt(z);
    Lambda = sqrt(k/h) * sqrt(z);


    dAlpha = 0.5*sqrt(h/k)*(1/sqrt(z))*dz;
    dLambda = 0.5*sqrt(k/h)*(1/sqrt(z))*dz;

    F = l + z;
    dF = dz;
```

```c

    E = Lambda;
    dE = dLambda;

    D = cosh(1/Lambda) + Alpha*sinh(1/Lambda);
    dD = (-0.1)*(1/(Lambda*Lambda))*(sinh(1/Lambda)+ \
    Alpha*cosh(1/Lambda))*dLambda + sinh(1/Lambda)*dAlpha;

    C = sinh(1/Lambda) + Alpha*cosh(1/Lambda);
    dC = (-0.1)*(1/(Lambda*Lambda))*(cosh(1/Lambda)+ \
    Alpha*sinh(1/Lambda))*dLambda + cosh(1/Lambda)*dAlpha;

    B = E/F;
    dB = (dE*F-E*dF)/(F*F);

    A = C/D;
    dA = (dC*D-C*dD)/(D*D);

    return dfx = dA*B + A*dB;
}
```

```c
int main()
{
    double xk, xk_old, diff, abserr,root, c;
    int k = 0;

    xk = 0.1;                              //Initial Guess
    xk_old = xk;
    c = 0.95;                              //Desired Thermal Efficiency

    abserr = 100;

    printf("Iteration         Root         \
         Thermal_Efficiency     Absolute Error    \n");
    while (abserr > MAXERR) {
        if ((therm_eff(xk)-c) == 0) {      // xk is the root
            root = xk;
            break;
        }else if ( derivative(xk) == 0 ) { // provide another initial guess
            printf("Provide another initial Guess\n");
            return 0;
        }else {                            // Update value of xk
            xk = xk - ((therm_eff(xk)-c)/derivative(xk));
            root = xk;
            k++;                           // Update iteration number
        }
```

```
        diff = xk - xk_old;
        abserr = sqrt(diff*diff);           // Determine the absolute error
        //abserr = sqrt((therm_eff(xk)-c)*(therm_eff(xk)-c));
        if (k<10) {
            printf("  ");
        }else if(k<100) {
            printf(" ");
        }
        printf("   %d          %0.9lf        %0.9lf          \
               %0.9lf\n",k,root,therm_eff(root),abserr);

        xk_old = xk;                       // Update the previous iteration value
    }

    printf("\nConverged at %0.9lf meters, after %d iterations\n",root,k);
    printf("\nThus the square cross section has\n");
    printf("Side Length of:  %0.9lf meters\n",root);
    printf("Area of       :  %0.9lf square meter\n",root*root);
    printf("Perimeter of  :  %0.9lf meters\n",root*4);

    return 0;
}
```

Solving Ideal Fluid Flow past a Cylinder Problems Using

LU Decomposition and SOR Method

Problem Statement

The potential at particular nodes of a fluid passing through a cylinder given the inflow velocity u_0 are to be computed using Direct methods (i.e LU Decomposition) and Iterative Method (i.e. Successive Over-Relaxation Method).

Results and Discussion

The governing equation for the fluid flow past a cylinder is given by fig. 4.

$$[A]\vec{x} = \vec{b}$$

where $[A] =$

$$\begin{bmatrix}
1.0250 & -0.4000 & 0 & 0 & -0.6250 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-0.4000 & 1.9135 & -0.2998 & 0 & 0 & -1.0338 & -0.1799 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -0.2998 & 2.1415 & -0.4417 & 0 & 0 & -1.4000 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -0.4417 & 1.0077 & 0 & 0 & 0 & -0.5660 & 0 & 0 & 0 & 0 & 0 \\
-0.6250 & 0 & 0 & 0 & 2.0500 & -0.8000 & 0 & 0 & -0.6250 & 0 & 0 & 0 & 0 \\
0 & -1.0338 & 0 & 0 & -0.8000 & 3.8097 & -0.4401 & 0 & 0 & -1.0000 & -0.0595 & -0.4764 & 0 \\
0 & -0.1799 & -1.4000 & 0 & 0 & -0.4401 & 4.0492 & -0.7067 & 0 & 0 & 0 & -1.1159 & -0.2067 \\
0 & 0 & 0 & -0.5660 & 0 & 0 & -0.7067 & 2.2161 & 0 & 0 & 0 & 0 & -0.9433 \\
0 & 0 & 0 & 0 & -0.6250 & 0 & 0 & 0 & 1.0250 & -0.4000 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1.0000 & 0 & 0 & -0.4000 & 2.0667 & -0.6667 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -0.0595 & 0 & 0 & 0 & -0.6667 & 1.3788 & -0.6526 & 0 \\
0 & 0 & 0 & 0 & 0 & -0.4764 & -1.1159 & 0 & 0 & 0 & -0.6526 & 2.5098 & -0.2650 \\
0 & 0 & 0 & 0 & 0 & 0 & -0.2067 & -0.9433 & 0 & 0 & 0 & -0.2650 & 1.4150
\end{bmatrix}$$

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \\ x_{13} \end{bmatrix} \qquad \vec{b} = u_0 \begin{bmatrix} 2 \\ 0 \\ 0 \\ 0 \\ 4 \\ 0 \\ 0 \\ 0 \\ 2 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

in which x_i is the potential at node i, and u_0 is the inflow velocity.

Figure 4: Equations of fluid past a cylinder based on finite-element idealization.

I.) Crout's Method:

The Algorithm for LU Decomposition is given by fig. 5.

$$u_{ii} = 1; \quad i = 1,2,\ldots,n$$

$$l_{ij} = \left\{ a_{ij} - \sum_{k=1}^{j-1} l_{ik} u_{kj} \right\}; \quad i \geq j; \; i = 1,2,\ldots,n$$

$$u_{ij} = \left\{ \frac{a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj}}{l_{ii}} \right\}; \quad i < j; \; j = 2,3,\ldots,n$$

Figure 5: Elements of the L and U Matrices Using Crout's Method.

A C Program (*see Appendix*) is created to implement the LU Decomposition and Forward and Backward substitution.

The results of LU Decomposition via Crout's Method are shown in fig. 6 for u_0 = 5 and fig. 7 for u_0 = 100.

```
Potential at Each node are:
x[1] = -199958.121300417
x[2] = -199981.446211289
x[3] = -199990.061189867
x[4] = -199983.336274209
x[5] = -199959.193357458
x[6] = -199985.783391902
x[7] = -199994.027736883
x[8] = -199978.088226071
x[9] = -199958.230170412
x[10] = -199981.725190652
x[11] = -199989.734500455
x[12] = -199998.277095927
x[13] = -199984.197588189
```

Figure 6: Results of Crout's-LU Decomposition with u_0 = 5.

```
Potential at Each node are:
x[1] = -3999162.426008333
x[2] = -3999628.924225785
x[3] = -3999801.223797331
x[4] = -3999666.725484178
x[5] = -3999183.867149163
x[6] = -3999715.667838034
x[7] = -3999880.554737666
x[8] = -3999561.764521423
x[9] = -3999164.603408235
x[10] = -3999634.503813033
x[11] = -3999794.690009100
x[12] = -3999965.541918535
x[13] = -3999683.951763778
```

Figure 7: Results of Crout's-LU Decomposition with u_0 = 100.

II.) Successive Over-Relaxation.

The [A] matrix is ill-conditioned since its determinant is very close to zeros. Using Microsoft excel I found that its determinant is **-0.008355299.** Hence partial pivoting is done to lessen the ill-conditioning.

A C Program (*see Appendix*) is created to implement the SOR method and preconditioning of the system. The relaxation factor is swept from 1 to 1.5 with 0.05 increments.

w	Total Iterations
1.00	14
1.05	21
1.10	25
1.15	30
1.20	35
1.25	40
1.30	46
1.35	52
1.40	60
1.45	69
1.50	83

Figure 8: Total iterations for each value of w of SOR method at $u_0 = 5$.

w	Total Iterations
1.00	14
1.05	23
1.10	27
1.15	32
1.20	37
1.25	42
1.30	48
1.35	55
1.40	63
1.45	75
1.50	90

Figure 9: Total iterations for each value of w of SOR method at $u_0 = 100$.

The total number of iterations for each value of w are shown in fig. 8 for u0 = 5 and fig. 9 for u0 = 100. Thus the optimum value of w is '1' or there should be no relaxation. The problem is repeated again to perform Gauss-Seidel (w=1) and it will yield the results shown in fig. 10 and 11.

```
Iteration number = 1  with tolerance = 199984.197588240
Iteration number = 2  with tolerance = 199989.267753028
Iteration number = 3  with tolerance = 186052.447665428
Iteration number = 12  with tolerance = 0.402511116
Iteration number = 13  with tolerance = 0.015622956
Iteration number = 14  with tolerance = 0.000000001

The answers are:
x_new[1] = -199958.121300465
x_new[2] = -199981.446211338
x_new[3] = -199990.061189916
x_new[4] = -199983.336274258
x_new[5] = -199959.193357507
x_new[6] = -199985.783391951
x_new[7] = -199994.027736933
x_new[8] = -199978.088226120
x_new[9] = -199958.230170461
x_new[10] = -199981.725190701
x_new[11] = -199989.734500504
x_new[12] = -199998.277095976
x_new[13] = -199984.197588238

Total Number of Iterations = 14
```

Figure 10: Results of SOR method with w = 1.0 and u_0 = 5

```
Iteration number = 1  with tolerance = 3999683.951764802
Iteration number = 2  with tolerance = 3999785.355060568
Iteration number = 3  with tolerance = 3721048.953308569
Iteration number = 12  with tolerance = 8.050222314
Iteration number = 13  with tolerance = 0.312459121
Iteration number = 14  with tolerance = 0.000000015

The answers are:
x_new[1] = -3999162.426009310
x_new[2] = -3999628.924226767
x_new[3] = -3999801.223798315
x_new[4] = -3999666.725485163
x_new[5] = -3999183.867150146
x_new[6] = -3999715.667839019
x_new[7] = -3999880.554738652
x_new[8] = -3999561.764522409
x_new[9] = -3999164.603409220
x_new[10] = -3999634.503814018
x_new[11] = -3999794.690010086
x_new[12] = -3999965.541919521
x_new[13] = -3999683.951764764

Total Number of Iterations = 14
```

Figure 11: Results of SOR method with w = 1.0 and u_0 = 100

Problems encountered

The generation of code for LU decomposition was troublesome because of the proper sequence in computing the L and U elements. For the SOR method I had problems for convergence because it is ill-conditioned. Thus I studied partial pivoting. I accomplished these problems by further reading.

References

- S. C. Chapra, R. P. Canale – Numerical Method for Engineers, 7th edition.

- K. J. Yap, ES 204 Notes – Numerical Solution of Simultaneous Algebraic Equations Part 1 and 2.

Appendix

Please see next pages. First 2 pages: C source code for LU Decomposition using Crout's Method. Last 3 pages: C source code for SOR method.